cssselect Documentation

Release 1.2.0

Simon Sapin

Apr 11, 2024

CONTENTS

cssselect is a BSD-licensed Python library to parse CSS3 selectors and translate them to XPath 1.0 expressions.

XPath 1.0 expressions can be used in lxml or another XPath engine to find the matching elements in an XML or HTML document.

Find the cssselect online documentation at https://cssselect.readthedocs.io.

Quick facts:

- Source, issues and pull requests on GitHub
- Releases on PyPI
- Install with `pip install cssselect`

Contents

CHAPTER

ONE

QUICKSTART

Use *HTMLTranslator* for HTML documents, *GenericTranslator* for "generic" XML documents. (The former has a more useful translation for some selectors, based on HTML-specific element types or attributes.)

```
>>> from cssselect import GenericTranslator, SelectorError
>>> try:
...     expression = GenericTranslator().css_to_xpath('div.content')
... except SelectorError:
...     print('Invalid selector.')
...
>>> print(expression)
descendant-or-self::div[@class and contains(concat(' ', normalize-space(@class), ' '), '␣
↪content ')]
```

The resulting expression can be used with lxml's XPath engine:

```
>>> from lxml.etree import fromstring
>>> document = fromstring('''
...   <div id="outer">
...     <div id="inner" class="content body">text</div>
...   </div>
... ''')
>>> [e.get('id') for e in document.xpath(expression)]
['inner']
```

CHAPTER

TWO

USER API

In CSS3 Selectors terms, the top-level object is a group of selectors, a sequence of comma-separated selectors. For example, `div, h1.title + p` is a group of two selectors.

`cssselect.parse`(*css: str*) → List[*Selector*]

Parse a CSS *group of selectors*.

If you don't care about pseudo-elements or selector specificity, you can skip this and use *`css_to_xpath()`*.

Parameters
`css` – A *group of selectors* as a string.

Raises
`SelectorSyntaxError` on invalid selectors.

Returns
A list of parsed *`Selector`* objects, one for each selector in the comma-separated group.

`class cssselect.Selector`

Represents a parsed selector.

`selector_to_xpath()` accepts this object, but ignores *`pseudo_element`*. It is the user's responsibility to account for pseudo-elements and reject selectors with unknown or unsupported pseudo-elements.

`canonical`() → str

Return a CSS representation for this selector (a string)

`pseudo_element`

	Selector	Pseudo-element
CSS3 syntax	`a::before`	`'before'`
Older syntax	`a:before`	`'before'`
From the Lists3 draft, not in Selectors3	`li::marker`	`'marker'`
Invalid pseudo-class	`li:marker`	`None`
Functional	`a::foo(2)`	`FunctionalPseudoElement(...)`

`specificity`() → Tuple[int, int, int]

Return the specificity of this selector as a tuple of 3 integers.

`class cssselect.FunctionalPseudoElement`(*name: str*, *arguments: Sequence[Token]*)

Represents selector::name(arguments)

`name`

The name (identifier) of the pseudo-element, as a string.

arguments

The arguments of the pseudo-element, as a list of tokens.

Note: tokens are not part of the public API, and may change between cssselect versions. Use at your own risks.

class cssselect.**GenericTranslator**

Translator for "generic" XML documents.

Everything is case-sensitive, no assumption is made on the meaning of element names and attribute names.

css_to_xpath(*css: str*, *prefix: str = 'descendant-or-self::'*) → str

Translate a *group of selectors* to XPath.

Pseudo-elements are not supported here since XPath only knows about "real" elements.

Parameters

- **css** – A *group of selectors* as a string.
- **prefix** – This string is prepended to the XPath expression for each selector. The default makes selectors scoped to the context node's subtree.

Raises

SelectorSyntaxError on invalid selectors, *ExpressionError* on unknown/unsupported selectors, including pseudo-elements.

Returns

The equivalent XPath 1.0 expression as a string.

selector_to_xpath(*selector: Selector*, *prefix: str = 'descendant-or-self::'*, *translate_pseudo_elements: bool = False*) → str

Translate a parsed selector to XPath.

Parameters

- **selector** – A parsed *Selector* object.
- **prefix** – This string is prepended to the resulting XPath expression. The default makes selectors scoped to the context node's subtree.
- **translate_pseudo_elements** – Unless this is set to `True` (as *css_to_xpath()* does), the *pseudo_element* attribute of the selector is ignored. It is the caller's responsibility to reject selectors with pseudo-elements, or to account for them somehow.

Raises

ExpressionError on unknown/unsupported selectors.

Returns

The equivalent XPath 1.0 expression as a string.

class cssselect.**HTMLTranslator**(*xhtml: bool = False*)

Translator for (X)HTML documents.

Has a more useful implementation of some pseudo-classes based on HTML-specific element names and attribute names, as described in the HTML5 specification. It assumes no-quirks mode. The API is the same as *GenericTranslator*.

Parameters

xhtml – If false (the default), element names and attribute names are case-insensitive.

2.1 Exceptions

exception cssselect.**SelectorError**

Common parent for *SelectorSyntaxError* and *ExpressionError*.

You can just use `except SelectorError:` when calling *css_to_xpath()* and handle both exceptions types.

exception cssselect.**SelectorSyntaxError**

Parsing a selector that does not match the grammar.

exception cssselect.**ExpressionError**

Unknown or unsupported selector (eg. pseudo-class).

CHAPTER

THREE

SUPPORTED SELECTORS

This library implements CSS3 selectors as described in the W3C specification. In this context however, there is no interactivity or history of visited links. Therefore, these pseudo-classes are accepted but never match anything:

- `:hover`
- `:active`
- `:focus`
- `:target`
- `:visited`

Additionally, these depend on document knowledge and only have a useful implementation in *`HTMLTranslator`*. In *`GenericTranslator`*, they never match:

- `:link`
- `:enabled`
- `:disabled`
- `:checked`

These applicable pseudo-classes are not yet implemented:

- `*:first-of-type`, `*:last-of-type`, `*:nth-of-type`, `*:nth-last-of-type`, `*:only-of-type`. All of these work when you specify an element type, but not with *

On the other hand, *cssselect* supports some selectors that are not in the Level 3 specification.

These parts of the Level 4 specification are supported (note that a large part of the Level 4 additions is not applicable to cssselect similarly to `:hover` or not representable in XPath 1.0 so the complete specification is unlikely to be implemented):

- The `:scope` pseudo-class. Limitation: it can only be used at a start of a selector.
- The `:is()`, `:where()` and `:has()` pseudo-classes. Limitation: `:has()` cannot contain nested `:has()` or `:not()`.

These are non-standard extensions:

- The `:contains(text)` pseudo-class that existed in an early draft but was then removed.
- The `!=` attribute operator. `[foo!=bar]` is the same as `:not([foo=bar])`.
- `:not()` accepts a *sequence of simple selectors*, not just single *simple selector*. For example, `:not(a.important[rel])` is allowed, even though the negation contains 3 *simple selectors*.

CHAPTER

FOUR

CUSTOMIZING THE TRANSLATION

Just like *HTMLTranslator* is a subclass of *GenericTranslator*, you can make new sub-classes of either of them and override some methods. This enables you, for example, to customize how some pseudo-class is implemented without forking or monkey-patching cssselect.

The "customization API" is the set of methods in translation classes and their signature. You can look at the source code to see how it works. However, be aware that this API is not very stable yet. It might change and break your sub-class.

CHAPTER

FIVE

NAMESPACES

In CSS you can use `namespace-prefix|element`, similar to `namespace-prefix:element` in an XPath expression. In fact, it maps one-to-one. How prefixes are mapped to namespace URIs depends on the XPath implementation.

CHAPTER

SIX

CHANGELOG

6.1 Version 1.2.0

Released on 2022-10-27.

- Drop support for Python 2.7, 3.4-3.6, add support for Python 3.7-3.11.
- Add type annotations (PEP 484 and PEP 561).
- More features from the CSS Selectors Level 4:
  - The `:is()` pseudo-class.
  - The `:where()` pseudo-class.
  - The `:has()` pseudo-class, with some limitations.
- Fix parsing `:scope` after a comma.
- Add parentheses to fix condition precedence in some cases.
- Private API changes related to the removal of the Python 2 support:
  - Remove `_unicode` and `_unichr` aliases from `csselect.parser`.
  - Remove `_basestring` and `_unicode` aliases from `csselect.xpath`.
  - Deprecate `csselect.xpath._unicode_safe_getattr()` and change it to just call `getattr()`.
- Include tests in the PyPI tarball.
- Many CI additions and improvements.
- Improve the test coverage.

6.2 Version 1.1.0

Released on 2019-08-09.

- Support for the `:scope` selector, which allows to access immediate children of a selector.
- Support for the `|E` syntax for type selectors without a namespace.
- A new selector method, `canonical`, returns the CSS expression of the selector, as a string.

6.3 Version 1.0.3

Released on 2017-12-27.

- Fix artifact uploads to pypi

6.4 Version 1.0.2

Released on 2017-12-26.

- Drop support for Python 2.6 and Python 3.3.
- Fix deprecation warning in Python 3.6.
- Minor cleanups.

6.5 Version 1.0.1

Released on 2017-01-10.

- Add support for Python 3.6.
- Documentation hosted on Read the Docs

6.6 Version 1.0.0

Released on 2016-10-21.

- Add code coverage reports.
- Fix `:nth-*(an+b)` pseudo-classes selectors. (except `*:nth-child()` which looks untranslatable to XPath 1.0.)

6.7 Version 0.9.2

Released on 2016-06-15.

- Distribute as universal wheel.
- Add support for Python 3.3, 3.4 and 3.5.
- Drop support for Python 2.5 as testing is getting difficult.
- Improve tests on pseudo-elements.

6.8 Version 0.9.1

Released on 2013-10-17.

- **Backward incompatible change from 0.9**: *selector_to_xpath()* defaults to ignoring pseudo-elements, as it did in 0.8 and previous versions. (*css_to_xpath()* doesn't change.)
- Drop official support for Python 2.4 and 3.1, as testing was becoming difficult. Nothing will break overnight, but future releases may on may not work on these versions. Older releases will remain available on PyPI.

6.9 Version 0.9

Released on 2013-10-11.

Add parser support for *functional pseudo-elements*.

Update: This version accidentally introduced a **backward incompatible** change: *selector_to_xpath()* defaults to rejecting pseudo-elements instead of ignoring them.

6.10 Version 0.8

Released on 2013-03-15.

Improvements:

- #22 Let extended translators override what XPathExpr class is used
- #19 Use the built-in `lang()` XPath function for implementing the `:lang()` pseudo-class with XML documents. This is probably faster than `ancestor-or-self::`.

Bug fixes:

- #14 Fix non-ASCII pseudo-classes. (Invalid selector instead of crash.)
- #20 As per the spec, elements containing only whitespace are not considered empty for the `:empty` pseudo-class.

6.11 Version 0.7.1

Released on 2012-06-14. Code name *remember-to-test-with-tox*.

0.7 broke the parser in Python 2.4 and 2.5; the tests in 2.x. Now all is well again.

Also, pseudo-elements are now correctly made lower-case. (They are supposed to be case-insensitive.)

6.12 Version 0.7

Released on 2012-06-14.

Bug fix release: see #2, #7 and #10 on GitHub.

- The tokenizer and parser have been rewritten to be much closer to the specified grammar. In particular, non-ASCII characters and backslash-escapes are now handled correctly.
- Special characters are protected in the output so that generated XPath exrpessions should always be valid
- The ~=, ^= and *= attribute operators now correctly never match when used with an empty string.

6.13 Version 0.6.1

Released on 2012-04-25.

Make sure that internal token objects do not "leak" into the public API and *Selector.pseudo_element* is an unicode string.

6.14 Version 0.6

Released on 2012-04-24.

- In `setup.py` use setuptools/distribute if available, but fall back on distutils.
- Implement the `:lang()` pseudo-class, although it is only based on `xml:lang` or `lang` attributes. If the document language is known from some other meta-data (like a `Content-Language` HTTP header or <meta> element), a workaround is to set a lang attribute on the root element.

6.15 Version 0.5

Released on 2012-04-20.

- Fix case sensitivity issues.
- Implement *HTMLTranslator* based on the HTML5 specification rather than guessing; add the `xhtml` parameter.
- Several bug fixes and better test coverage.

6.16 Version 0.4

Released on 2012-04-18.

- Add proper support for pseudo-elements
- Add specificity calculation
- Expose the *parse()* function and the parsed *Selector* objects in the API.
- Add the *selector_to_xpath()* method.

6.17 Version 0.3

Released on 2012-04-17.

- Fix many parsing bugs.
- Rename the `Translator` class to *GenericTranslator*
- There, implement `:target`, `:hover`, `:focus`, `:active` `:checked`, `:enabled`, `:disabled`, `:link` and `:visited` as never matching.
- Make a new HTML-specific `HTMLTranslator` subclass. There, implement `:checked`, `:enabled`, `:disabled`, `:link` and `:visited` as appropriate for HTML, with all links "not visited".
- Remove the `css_to_xpath` function. The translator classes are the new API.
- Add support for `:contains()` back, but case-sensitive. lxml will override it to be case-insensitive for backward-compatibility.

Discussion is open if anyone is interested in implementing eg. `:target` or `:visited` differently, but they can always do it in a `Translator` subclass.

6.18 Version 0.2

Released on 2012-04-16.

- Remove the `CSSSelector` class. (The `css_to_xpath()` function is now the main API.)
- Remove support for the `:contains()` pseudo-class.

These changes allow cssselect to be used without lxml. (Hey, this was the whole point of this project.) The tests still require lxml, though. The removed parts are expected to stay in lxml for backward-compatibility.

`:contains()` only existed in an early draft of the Selectors specification, and was removed before Level 3 stabilized. Internally, it used a custom XPath extension function which can be difficult to express outside of lxml.

- Separate the XPath translation from the parsed objects into a new `Translator` class.

Subclasses of `Translator` can be made to change the way that some selector (eg. a pseudo-class) is implemented.

6.19 Version 0.1

Released on 2012-04-13.

Extract lxml.cssselect from the rest of lxml and make it a stand-alone project.

Commit `ea53ceaf7e44ba4fbb5c818ae31370932f47774e` was taken on 2012-04-11 from the 'master' branch of lxml's git repository. This is somewhere between versions 2.3.4 and 2.4.

The commit history has been rewritten to:

- Remove lxml files unrelated to cssselect
- Import the early history from the 'html' branch in the old SVN repository
- Fix author names in commits from SVN

This project has its own import name, tests and documentation. But the code itself is unchanged and still depends on lxml.

6.20 Earlier history

Search for *cssselect* in lxml's changelog

PYTHON MODULE INDEX

C

INDEX